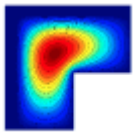
MATLAB news & notes



# MATLAB Incorporates LAPACK

## Increasing the speed and capabilities of matrix computation

by Cleve Moler

MATLAB started its life in the late 1970s as an interactive calculator built on top of LINPACK and EISPACK, which were then state-of-the-art Fortran subroutine libraries for matrix computation. The mathematical core for all versions of MATLAB, up to version 5.3, has used translations to C of about a dozen of the Fortran subroutines from LINPACK and EISPACK.

LAPACK is the modern replacement for LINPACK and EISPACK. It is a large, multi-author, Fortran library for numerical linear algebra. A new version was released in July and is available from NETLIB (www.netlib.org/lapack). LAPACK was originally intended for use on supercomputers and other high-end machines. It uses block algorithms, which operate on several columns of a matrix at a time. On machines with high-speed cache memory, these block operations can provide a significant speed advantage. LAPACK also provides a more extensive set of capabilities than its predecessors do.

**Current Issue**
Winter 2001
**Cleve's Corners**
1994-2001
**Previous Issues**
Winter 2000
Summer 1999
Winter 1999
Summer 1998
Winter 1998
MATLAB Special Edition 1997
Simulink Special Edition 1997
Fall 1996
Summer 1996
Spring 1996
Winter 1996
**Subscribe Now**

The speed of all these packages is closely related to the speed of the Basic Linear Algebra Subroutines, or BLAS. EISPACK did not use any BLAS. LINPACK used Level 1 BLAS, which operate on only one or two vectors, or columns of a matrix, at a time. Until now, MATLAB has used carefully coded C and assembly language versions of these Level 1 BLAS. LAPACK's block algorithms also make use of Level 2 and Level 3 BLAS, which operate on larger portions of entire matrices. The NETLIB distribution of LAPACK includes Reference BLAS written in Fortran. But the authors intended that various hardware and operating system manufacturers provide highly optimized, machine-specific, versions of the BLAS for their systems.

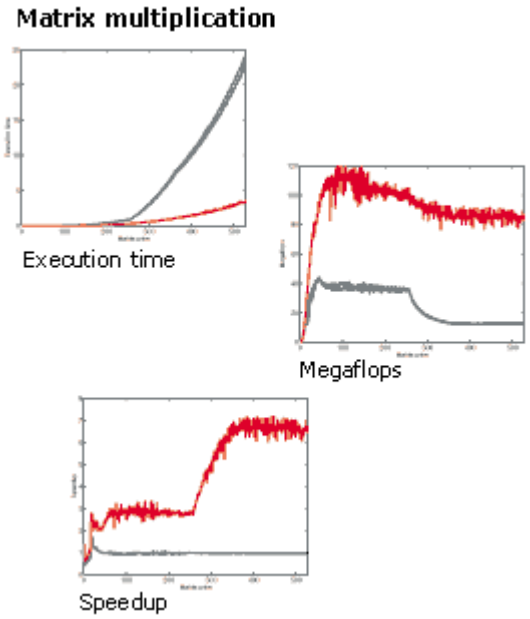
It is finally time to incorporate LAPACK into MATLAB. Almost all modern machines have enough cache memory to profit from LAPACK's design. Several key chip and computer vendors now offer optimized Level 1, 2, and 3 BLAS. A new alternative to the vendor BLAS is available from ATLAS, a research project at the University of Tennessee, where routines optimized for any particular machine can be generated automatically from parameterized code fragments.

The first MATLABs ran in the one-half megabyte memory available on the first PC, so it was necessary to keep code size at a minimum. One general-purpose eigenvalue routine,a single-shift complex QZ algorithm not in LINPACK or EISPACK, was developed for all complex and generalized eigenvalue problems. The extensive list of functions now available with LAPACK means that MATLAB's space saving general-purpose codes can be replaced by faster, more focused routines. There are now 16 different code paths underlying the `eig` function, depending on whether there are one or two arguments, whether the arguments are real or complex, whether the problem is symmetric and whether the eigenvectors are requested.

Measuring the speed improvements provided by LAPACK-based MATLAB is a delicate and interesting business. Operations on small matrices happen so fast that the time is hard to measure accurately. To see the effects of cache usage patterns, it is desirable to use all values of the matrix order, *n*, up to several hundred. As *n* increases, the execution time increases as $n^3$, so one careful experiment can require several hours to run.

With MATLAB version 5.3, which is currently available on Release 11 of the MathWorks CD, the code for matrix computation is contained in a library called `numerics.dll` on the PC and `libmwnumerics.so` on various UNIX machines. We have measured execution times and speed improvements for three different versions of this library and color-coded the results in our graphs. The three versions are:

- Current (blue): The numerics library included with MATLAB version 5.3
- Reference (green): A new numerics library based on LAPACK and machine-independent Fortran Reference BLAS
- Optimized (red): A new numerics library based on LAPACK and BLAS optimized for each particular computer



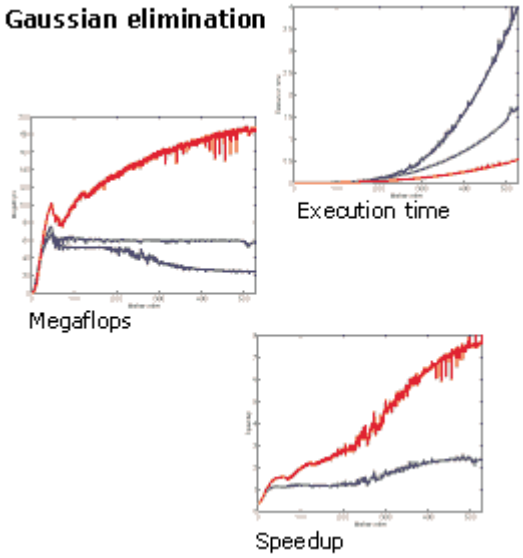

Our first three graphs show the results of timing experiments involving matrix multiplication, run on a fairly old (and slow) Sun SPARC 10 with the Solaris operating system. For each value of *n* from 1 to 528, we generated two *n*-by-*n* matrices and then used `tic` and `toc` to time the matrix product `C = A*B`. For small values of *n*, we did this several thousand times to avoid clock resolution difficulties. The first graph shows the execution times for our three versions of the numerics library. The current version and the version with the reference BLAS use unblocked algorithms in this situation and run at nearly the same speed. They require almost 25 seconds to compute a matrix product of order 528. The optimized version with the BLAS provided by Sun in their *Sun Performance Library* does the computation in less than 3.6 seconds.

Using the fact that matrix multiplication requires $2n^3$ floating-point operations, we obtain the megaflop rates shown in the second graph. For values of *n* larger than 256, the megaflop rate for the current and reference libraries drops from about 40 to about 12. (You can also see the the kink in the green and blue execution time graph near *n*=256.) This is because there are three matrices involved in the computation, and when their order is larger than 256 they no longer fit in the cache memory of this particular computer. Cache size also has some effect on the red line, but it is not so severe. The third graph shows the ratio of the execution times. This is the speedup we can expect from a block algorithm for matrix multiplication in this environment. We get a factor of almost three for matrices that fit in cache and a factor between six and seven for larger matrices.

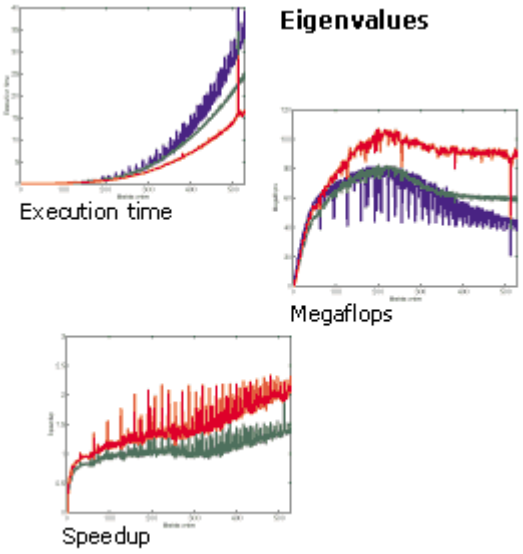
The irregular detailed behavior of our graphs is only partly due to experimental noise in the timings. There are also complicated, but reproducible, effects of what we might call "cache resonance," interactions between the memory access patterns of a particular algorithm, the locations in memory of the matrices involved and cache line size and replacement policy. For example, values of *n* that are divisible by a power of two sometimes lead to a lower megaflop rate.

Multiplication of large matrices is rarely the rate determining operation in serious applications, so matrix multiplication, by itself, is an overly simplistic benchmark. But matrix multiplication is the

most important routine in the Level 3 BLAS because it is the heart of the more complicated block algorithms in LAPACK itself.




Our next three graphs show the performance of Gaussian elimination, as seen in the function `lu(A)`, on a 400 MHz Pentium PC running Linux. Again, *n* takes on all values from 1 to 528 and the blue, green and red lines correspond to the current numerics library, our new library with reference BLAS, and our new library with optimized BLAS. The three graphs show execution time, megaflops, and speedup. For Linux, we are using the optimized BLAS developed by Greg Henry of the University of Tennessee and Intel Corp. The effect of operating out of cache is not so dramatic here because there is only one matrix and the working order of the matrix decreases as the triangular decomposition proceeds. On this machine, we reach almost 200 megaflops with the optimized BLAS. For large matrices, we can expect `lu(A)` to be up to eight times faster.

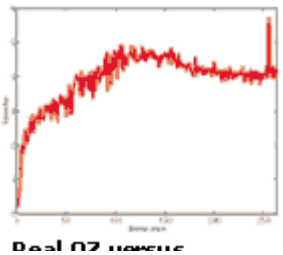



The next three graphs show performance of eigenvalue computation, as done by `eig(A)`, on a 233 MHz Pentium laptop running Windows NT. Optimized BLAS for this machine are available from Intel's Math Kernel Library and from the ATLAS project. These two sets of BLAS have roughly equal performance. For the eigenvalue computation, the speed improvement factor approaches two, because there is not so much opportunity for block algorithms. This is certainly significant, but not as spectacular as the speedups for the simpler algorithms.

Our final graph involves a different kind of comparison. The graph still shows speedup as a function of matrix order, but the comparison is between two different algorithms rather than between two implementations of essentially one algorithm. The primary tool for the solution of generalized matrix eigenvalue problems,

$$Ax = \lambda Bx$$

is the function `qz(A,B)`. MATLAB 5.3 has only one version of the QZ algorithm. It always uses complex arithmetic and produces complex results, even if the input matrices are real. LAPACK provides a real QZ algorithm for real matrices, so we can now have `qz(A,B,'real')` and `qz(A,B,'complex')`. The graph shows the ratio of the execution for these two calculations. For large real matrices, the real QZ algorithm is four times faster than the complex QZ algorithm. The big spike at *n* = 256 is a cache resonance phenomenon.



LAPACK with machine-specific BLAS will eventually offer the opportunity to use multithreading for additional speed enhancements. Some of the high-end PCs and workstations available today have two, or even four, processors sharing a common memory. It will be possible with multithreaded BLAS to devote all of the processors to a single matrix computation. But in the MATLAB environment, it might be more effective to use multiple processors for other activities, like graphics. We have not yet done any serious investigation of these tradeoffs.

Regrettably, one popular MATLAB feature must be a casualty with the introduction of LAPACK. `flops` function, which keeps a running count of the number of floating-point operations, is no longer feasible. Most of the floating point operations are now done in optimized BLAS that do not keep flop counts. However, with modern computer architectures, floating-point operations are no longer the dominant factor in execution speed. Memory references and cache usage are most important.

An LAPACK-based numerics library will be part of the next major release of MATLAB. In the meantime, if you want your own copy for MATLAB 5.3, please visit our Web page,

www.mathworks.com/company/newsletter/clevescorner/win00.cleve.shtml

You will find numerics libraries available for several different computers. If you install them on your machine, you should see computations involving large matrices speed up by at least a factor of two and possibly by a factor as large as eight.

related topics:

Using MathWorks Products For... | Training | MATLAB Based Books | Third-Party Products